Ruby Classes

You will learn how classes are implemented in the Ruby language and how they are different.

Our progress so far:

- ~~Open classes~~
- ~~Duck Typed~~
- ~~Just a runnable code~~
- ~~Object themselves~~*
- Scopes
- Modules with hierarchies
- Inheritance hierarchy
- Ruby object model
- Method lookup
- Etc.

Insight 7:

Classes are Modules with hierarchy

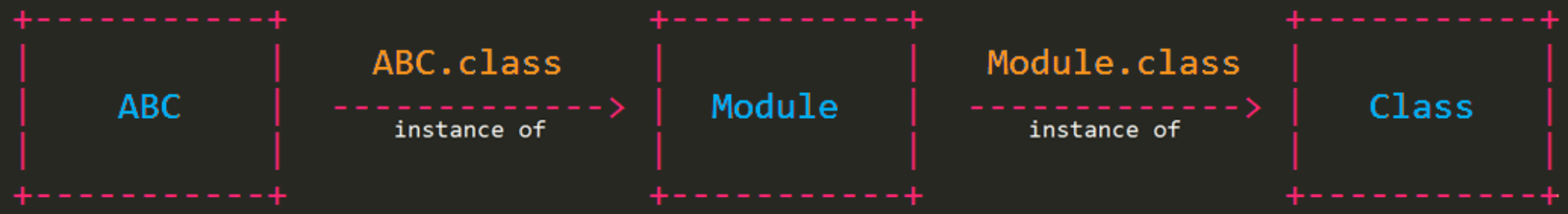
Classes are modules with 3 extra methods: new, allocate & superclass.
These methods allows us to create instances and have an inheritance hierarchy.

Classes are modules

```ruby
Class.superclass                                    # Module

Class.instance_methods - Module.instance_methods    # [:allocate, :new, :superclass]
```

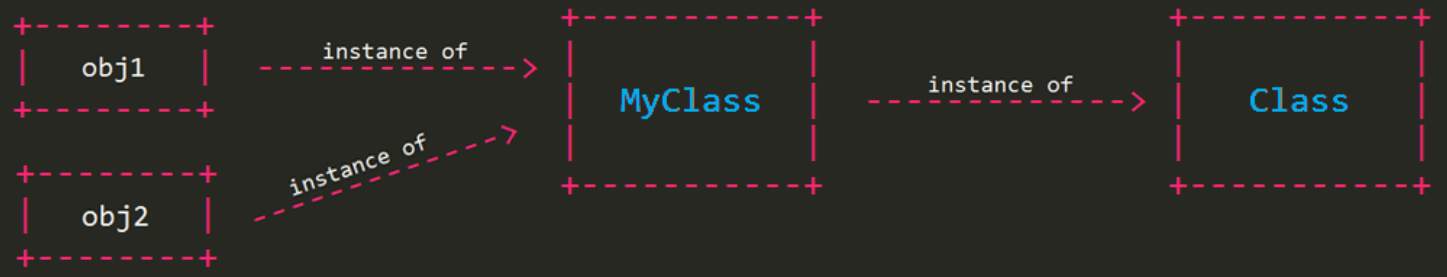
```ruby
module ABC
end

ABC = Module.new
```

```ruby
ABC.class                                           # Module
Module.class                                        # Class
```

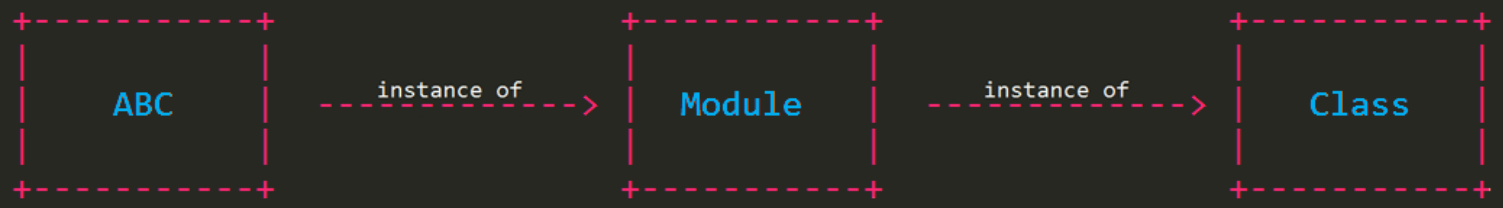


Comparing classes to modules

```ruby
class MyClass
end

obj1 = MyClass.new
obj2 = MyClass.new

obj1.class          # MyClass
MyClass.class       # Class
```




```ruby
module ABC
end

ABC.class           # Module
Module.class        # Class
```



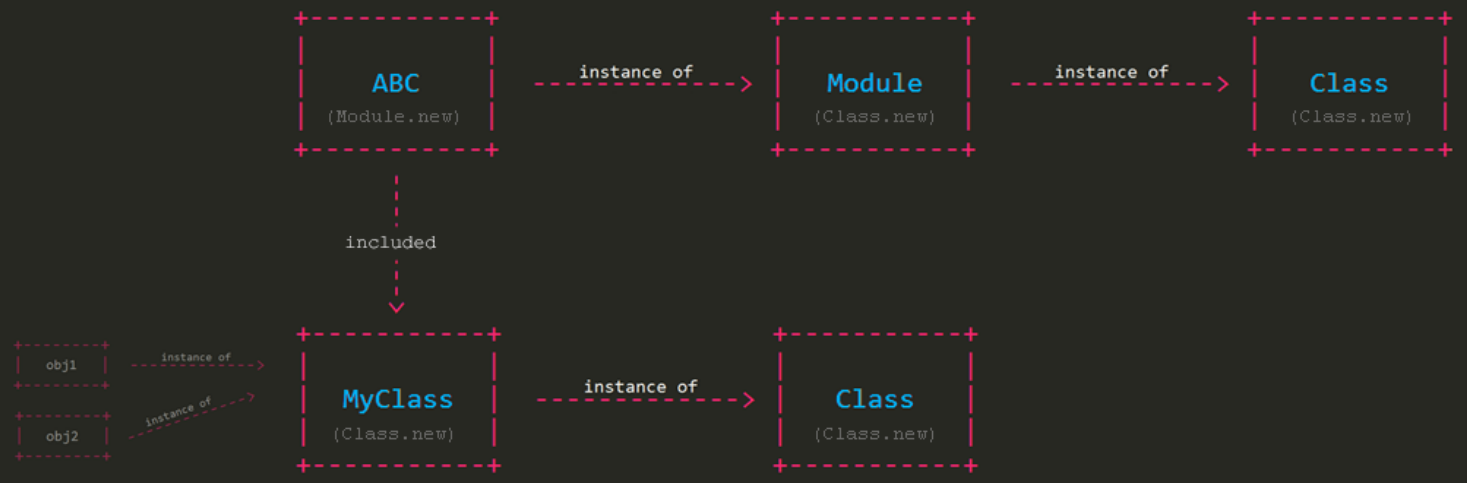
Including modules into classes

```ruby
module ABC
    def module_method
        @instance_var = "changed by module method"
    end
end


class MyClass
    include ABC

    def initialize
        @instance_var = "initialized in MyClass"
    end

    def instance_var
        @instance_var   # attr_reader :instance_var
    end
end
```

```ruby
obj = MyClass.new

obj.instance_var          # "initialized in MyClass"
obj.module_method
obj.instance_var          # "changed by module method"
```

Including modules into classes (cont.)

```ruby
module ABC
    @module_var = "module var"

    def self.module_var
        @module_var
    end
end


class MyClass
    include ABC

    @class_var = "class var"

    def self.module_var2
        @module_var
    end
end
```




```ruby
ABC.module_var
MyClass.module_var2
MyClass.module_var
```

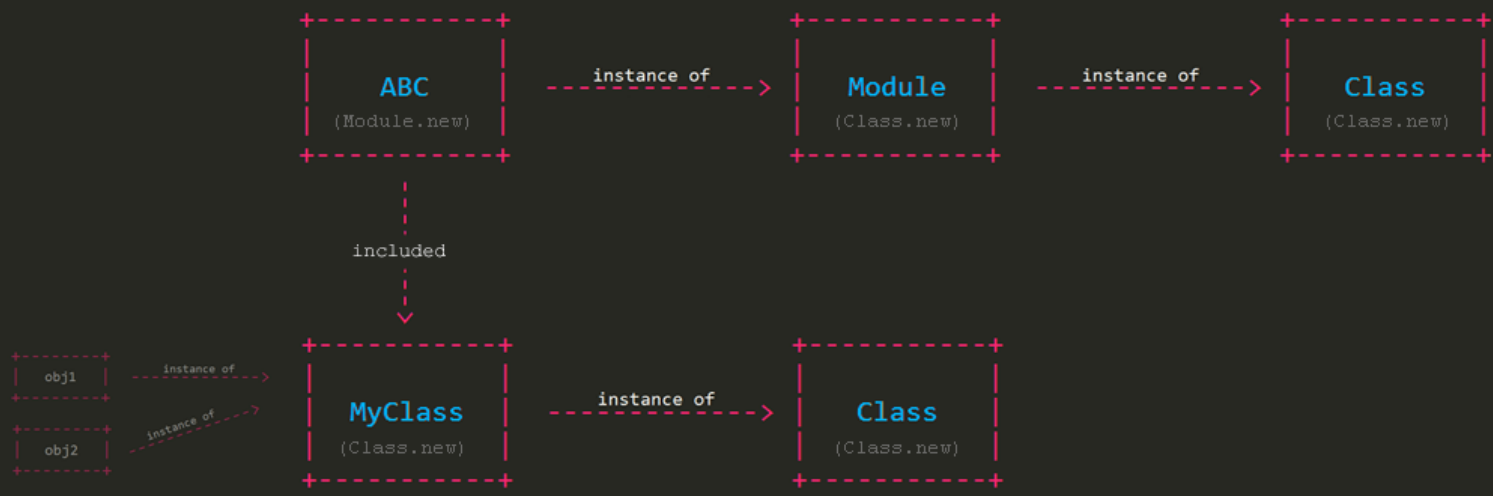
```ruby
# "module var"
# nil
# NoMethodError: undefined method `module_var' for MyClass
```

Defining class methods using modules

```ruby
module ABC
    def foo
        "Hurray..."
    end
end
```

```ruby
class MyClass
    class << self
        include ABC
    end
end
```

```ruby
class MyClass

    extend ABC

end
```




```ruby
ABC.foo
MyClass.foo
```

```ruby
# NoMethodError: undefined method `foo' for ABC:Module
# "Hurray..."
```

Recap:

- Classes are Modules that you can initialize and have inheritance hierarchy
- Everything we've learnt about classes also apply to modules

Our progress so far:

- ~~Open classes~~
- ~~Duck Typed~~
- ~~Just a runnable code~~
- ~~Object themselves~~*
- ~~Scopes~~
- ~~Modules with hierarchies~~
- Inheritance hierarchy
- Ruby object model
- Method lookup
- Etc.